SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

DIPLOMSKI RAD

Siniša Španić

Zagreb, 2015.

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

DIPLOMSKI RAD

Mentor:

Prof. dr. sc. Andrej Jokić

Student:

Siniša Španić

Zagreb, 2015.

Izjavljujem da sam ovaj rad izradio samostalno koristeći stečena znanja tijekom studija i navedenu literaturu.

Zahvaljujem se mentoru prof. dr. sc. Andreju Jokiću i profesoru dr. sc. Mladenu Crnekoviću na ukazanoj pomoći tijekom rada

Siniša Špani

SVEUČILIŠTE U ZAGREBU

FAKULTET STROJARSTVA I BRODOGRADNJE

Središnje povjerenstvo za završne i diplomske ispite

Povjerenstvo za diplomske ispite studija strojarstva za smjerove:
proizvodno inženjerstvo, računalno inženjerstvo, industrijsko inženjerstvo i menadžment, inženjerstvo materijala i mehatronika i robotika

Sveučilište u Zagrebu Fakultet strojarstva i brodogradnje	
Datum	Prilog
Klasa:	
Ur.broj:	

DIPLOMSKI ZADATAK

Student: **SINIŠA ŠPANIĆ** Mat. br.: 0035162660

Naslov rada na hrvatskom jeziku: **VIZUALNO PREPOZNAVANJE I MODELIRANJE RADNOG PROSTORA MOBILNOG ROBOTA**

Naslov rada na engleskom jeziku: **VISUAL RECOGNITION AND MODELING OF MOBILE ROBOT WORKING SPACE**

Opis zadatka:

Uređenost okoline mobilnog robota ključna je značajka za efikasno izvršenje zadatka. Kako se okolina često ne može prilagoditi potrebi robota, potrebno ju je što bolje spoznati. Jedan od efikasnih načina, koji ne zahtijeva direktnu interakciju robota s okolinom, je postavljanje kamere (nepokretne ili mobilne) iznad robota s ciljem snimanja i prepoznavanja objekata (prepreka) i samog robota.

Na Katedri za strojarsku automatiku nalazi se poligon s mobilnim robotima i kamera visoke razlučivosti montirana iznad poligona koja snima robote i prepreke. Potrebno je izraditi modularnu aplikaciju koja će prepoznati prepreke i robota, a informaciju o tome preko virtualnog komunikacijskog kanala prenijeti drugoj aplikaciji u dogovorenom obliku. Aplikacija treba raditi u realnom vremenu.

U radu je potrebno:

- definirati algoritam koji će razlikovati robote od prepreka i utvrditi njihov položaj,
- podatke o objektima slati drugoj aplikaciji,
- prikazati rezultat u pomoćnoj aplikaciji.

Zadatak zadan:	Rok predaje rada:	Predviđeni datum obrane:
24. rujna 2015.	26. studenog 2015.	2., 3. i 4. prosinca 2015.

Zadatak zadao: Predsjednik Povjerenstva:

Prof.dr.sc. Andrej Jokić

Prof. dr. sc. Franjo Cajner

SADRŽAJ

POPIS SLIKA

POPIS TABLICA

SAŽETAK

Ovaj rad načinjen je sa ciljem vizualnog prepoznavanja eMIR robota u njegovom radnom prostoru pomoću kamere montirane iznad radnog prostora. Informacije dobivene vizualno šalju se zatim preko virtualnog komunikacijskog kanala drugoj aplikaciji koja ih zatim prikaže. Glavni i pomoćni program napravljeni se u Qt programskom okruženju i korištenjem biblioteka za računalni vid OpenCV. Glavni program dohvaća slike s kamere montirane iznad radnog prostora, traži eMiR robota u zadanoj boji te iz položaja robota na slici uzima informacije o njegovom položaju koje zatim pretvara u dogovoreni format te šalje preko virtualnog komunikacijskog kanala. Na drugoj strani komunikacijskog kanala pomoćni program prihvaća te informacije i prikazuje ih. Prepoznavanje robota napravljeno je na temelju razlika u njihovoj boji. Testovima je pokazan utjecaj promjene svjetlosnih uvjeta na detekciju objekta. Testiranjem je pokazano da prepoznavanje robota funkcionira u konstantnim svjetlosnim uvjetima za koje se sustav baždari.

SUMMARY

This work is made with the goal of the visual recognition eMiR mobile robot in his working space with help of the camera mounted above the working space. Information obtained visually are sent over the virtual communication channel to another application that then displays them. Main and auxiliary applications are made in Qt programming environment and with using the library for computer vision OpenCV. Main program receives camera images search eMiR robot and from the position of the robot on the picture calculates its position. Informations obtained are then translated into agreed format and are sent over via virtual communication channel. Auxiliary application reads those information obtained from communication channel and displays them. Visual recognition of the eMiR robot is made on the basis of differences in their color. Tests showed the influence of changing light conditions on the detection of the object. Also tests showed that visual recognition of the eMiR robot works in the constant light conditions for which is calibrated.

1. UVOD

Za uspješnost izvršenje zadatka robot mora poznavati svoje mjesto u prostoru te imati informacije o sebi i okolini. Da bi se dobile tražene informacije roboti se opremaju senzorima. U ovom slučaju potrebno je preko kamere montirane iznad poligona razlučiti robota od okoline i definirati mu položaj u odnosu na središte poligona, tj. na točku iznad koje se nalazi kamera, te dobivene informacije proslijediti dalje preko zadanog komunikacijskog kanala te istodobno primati naredbe od drugih programa. Informacije o robotu moguće je dobiti samo preko kamere montirane iznad poligona te nije dopuštena direktna komunikacija s robotom. Nepostojanje bilo kakve povratne veze pomoću koje bi bilo moguće provjeriti izračunate informacije stavlja visok zahtjev na točnost tih informacija. Da bi bilo moguće uspješno razlikovanje robota od okoline potrebno je na robotu pronaći jednoznačne značajke kakvih nema u okolini. Jedna od takvih najjednostavnijih značajki je boja pošto su roboti na kojima se radi crvene, plave i žute boje. U radu će se prepoznavanje robota raditi na principu njihovih različitih boja.

2. EDUKACIJSKI MOBILNI ISTRAŽIVAČKI ROBOT(eMIR)

2.1 Mehanički sustav

Napravljena su tri identična robota [Slika 1], obojana u plavu, crvenu i žutu boju. Gibanje robota je realizirano korištenjem diferencijalnog modela. Osnovne dimenzije robota su duljina 300 mm, širina 250 mm i visina 110 mm. Roboti su napravljeni od pleksiglasa, donja ploča je debljine 10 mm, a gornja 6 mm. Sve komponente su zaštićene između te dvije ploče. Masa robota je 3,5 kg i pogoni ga standardna 12V/2 Ah punjiva baterija koja mu omogućuje 3 do 4 sata autonomije.

Slika 1. eMIR roboti

Kinematički model unutarnjih i vanjskih brzina mobilnog robota određen je sljedećim jednadžbama:

$$\boldsymbol{v = \frac{D(\omega_R + \omega_L)}{4}} \quad (1)$$

$$\boldsymbol{\omega = \frac{D(\omega_R - \omega_L)}{2 * L}} \quad (2)$$

v - translacijska brzina

ω -rotacijska brzina

ω_R i ω_L- kružne brzine desnog i lijevog kotača

D=80 mm- promjer kotača

L=240 mm- udaljenost između kotača

Uzevši to sve u obzir, uz maksimalnu brzinu motora, najviša brzina robota je oko 185 mm/s, a najviša rotacijska brzina iznosi 90 °/s.

2.2 Energetski sustav

Robot je pogonjen sa 2 istosmjerna elektromotora sa mehaničkim mjenjačkim omjerom od 1:71. Motori daju maksimalni moment od 0,14 Nm. Za upravljanje elektromotora koristi se integrirani sklop LMD18200[1]. Taj sklop sadrži sve potrebne zaštitne funkcije, što ima za posljedicu manje dijelova na robotu. Maksimalna struja na bateriji iznosi 450 mA sa svim senzorima i uključenom kamerom.

[1] Integrirani sklop za kontrolu i upravljanje kretanjem kojeg proizvodi Texas Instruments

2.3 Senzorski sustav

Senzori ugrađeni u motore koriste se za mjerenje brzine svakog kotača. Svaki enkoder ima 2 kanala sa po 7 impulsa. To daje robotu rezoluciju od 0,72 º po pulsu, što u translacijskog gibanju označava 0,5 mm. Za vrijeme rada mjeri se napon baterije i koristi se u algoritmu kojim robot odlučuje o daljnjim radnjama. Robot je opremljen sa 6 infracrvenih senzora tipa GP2Y0A21[2] sa mjernim područjem od 10 do 80 cm. Ti senzori imaju nelinearnu karakteristiku koja se interno preračunava u centimetre. Mjerenje udaljenosti odvija se svakih 10 ms. Položaj senzora na robotu nije ekvidistantan, nego se pokušava oponašat ljudska percepcija. Pošto infracrveni senzori ne mjere udaljenosti manje od 10 cm, dodani su i digitalni senzori GP2Y0D810Z0F[3] koji mjere udaljenosti od 2 do 10 cm.

2.4 Upravljački sustav

Središte upravljačkog sustava robota je Atmelov mikrokontroler AT89C51ID2 sa 64kB memorije i 1 MHz frekvencijom te je to sasvim dovoljno za funkcioniranje robota. Bežična komunikacija odvija se bluetooth modulom Sparkfun WRL-00582 sa brzinom prijenosa podataka od 57600bps. Mikrokontroler nema ugrađeni analogno-digitalni konverter te se koristi vanjski konverter ADC LTC1294[4]. Napajanje robota se kontrolira sa bistabil relejom, koji se može ugasiti računalno. Paralelno sa mikrokontrolerom postavljen je pomoćni kontroler AT89C4051[5] koji nadzire komunikaciju. 8 ulazno/izlaznih linija na glavnom mikrokontroleru je slobodno i može biti korišteno za dodatne senzore. Također moguća je signalizacija preko ugrađene sirene.

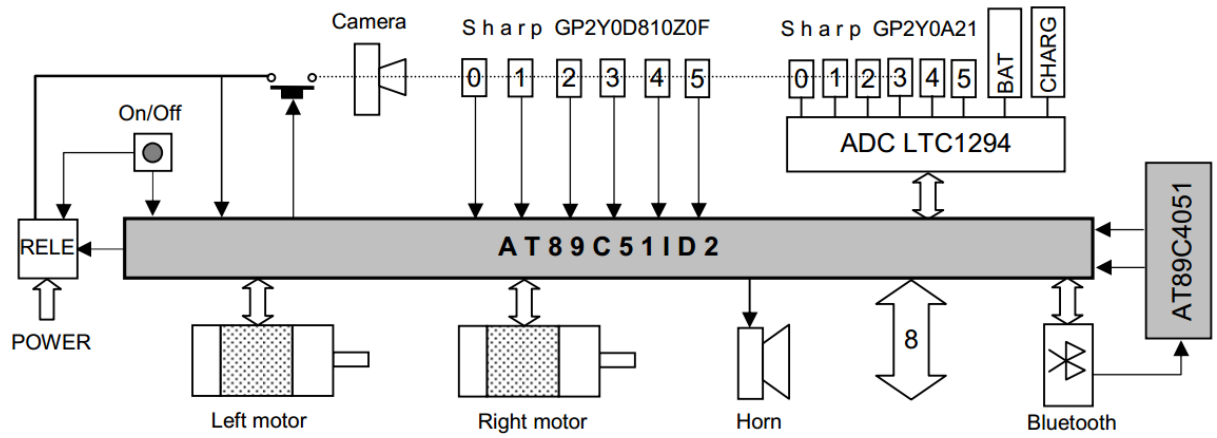
Cijeli upravljački sustav robota koristi samo 3,5 kB memorije, što je oko 5% dostupne memorije na mikrokontroleru.

[2] Sharpov infracrveni senzor sa područjem rada od 10 do 80cm
[3] Sharpov digitalni senzor koji mjeri udaljenosti od 2 do 10cm
[4] Analogno digitalni konverter kojeg proizvodi Linear Technology
[5] Atmelov 8-bitni mikrokontroler

Upravljački program pruža sljedeće usluge:

- Nadzor komunikacije i primanje naredbi
- Kontroliranje i izvršavanje naredbi
- Mjerenje brzine elektromotora
- Upravljanje brzinom motora
- Čitanje jednog infracrvenog senzora
- Mjerenje razine energije u bateriji.

Cijeli upravljački sustav robota [*Slika 2*] izvodi se na računalu i naredbe kretanja se šalju robotu. Ako je robot u kretanju i ne primi naredbu unutar 1 sekunde zaustavlja se. Ako nema naredbe 120 sekundi robot se gasi.



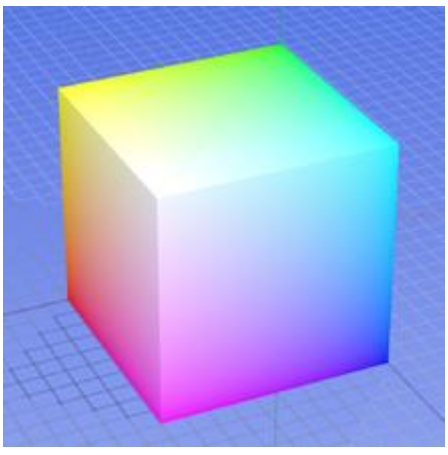
Slika 2. Nacrt upravljačkog sustava

3. MODEL BOJE

Model boje u računalnoj grafici odgovara specifikaciji trodimenzionalnog koordinatnog sustava i vidljivog podskupa u tom koordinatnom sustavu u kojem se nalaze sve boje iz određenog područja boja. Svrha modela boje je prikladan način specifikacije boja iz određenog područja boja. Razvijene su dvije skupine modela boja. Prva skupina obuhvaća modele koji su sklopovski orijentirani. Primjeri sklopovski orijentiranih modela boje su RGB (Red-Green-Blue), te CMY (Cyan-Magenta-Yellow) i CMYK (Cyan-Magenta-Yellow-Black). Druga skupina modela obuhvaća korisnički orijentirane modele koji su bliže načinu raspoznavanja svojstava boja od strane korisnika. Primjeri takvih modela su HSV (Hue-Saturation-Value), HLS (Hue-Lightness-Saturation) i HVC (Hue-Value-Chromaticity).

3.1 RGB model

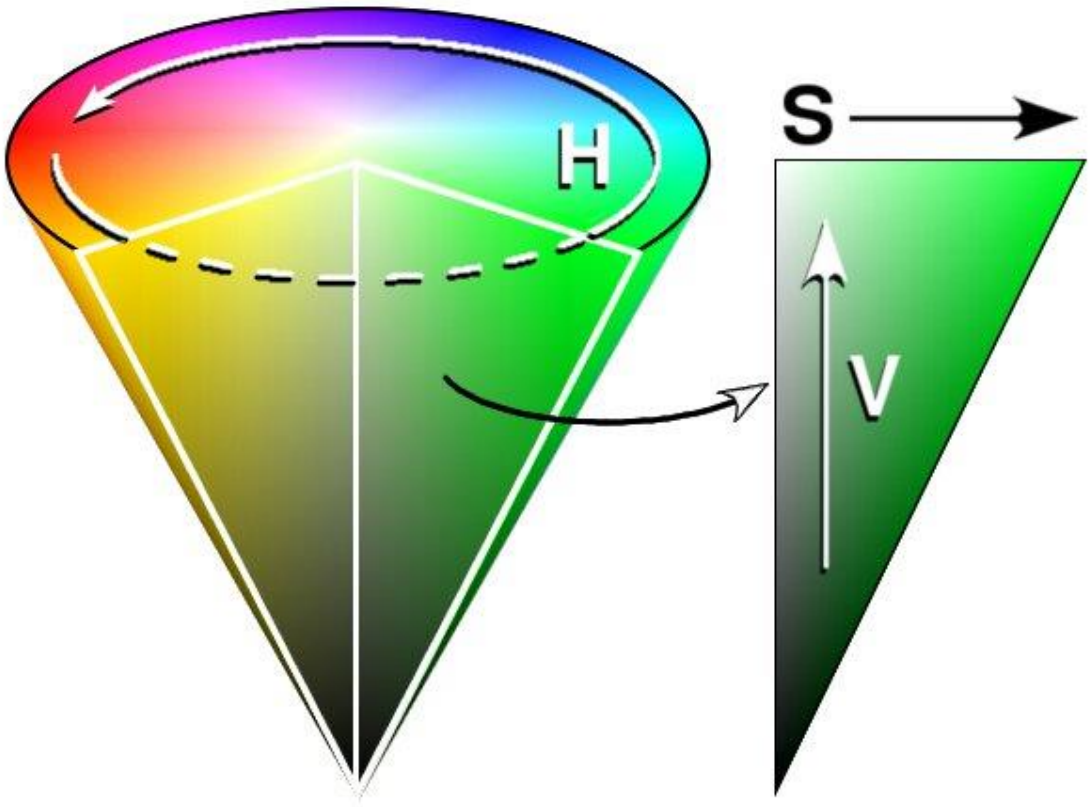
RGB model boje često se koristi u monitorima u boji i rasterskoj grafici. Taj model može se geometrijski definirati kao kocka [*Slika 3*] gdje se vrijednosti komponenata (Crvena-Zelena-Plava) koriste kao kartezijske koordinate u euklidskom prostoru. Za RGB model koriste se ne negativne koordinate između vrijednosti 0 i 1 gdje se crna boja stavlja u ishodište (0,0,0), a bijela na suprotni kut s koordinatama 1,1,1.



Slika 3. RGB geometrijski prikaz

3.2 HSV model

HSV model boje [*Slika 4*] bliži je intuitivnom shvaćanju boje. Ovaj model definira se u cilindričnom koordinatnom sustavu. Boje su prikazane u podskupu prostora omeđenom šesterostranom piramidom. Vrh piramide je u ishodištu i odgovara crnoj boji. Vertikalna os V određuje sjajnost boje (value). Baza piramide je na razini V=1 što odgovara skupu sjajnih boja. Nijansa (hue) određena je kutem zakreta H oko vertikalne osi V. Kutovi zakreta komplementarnih boja razlikuju se za 180°. radijalna udaljenost od osi V određuje razinu zasićenosti boje (saturation).



Slika 4. HSV model boja

HSV model boja se sastoji od 3 glava dijela: nijanse (hue), zasićenosti (saturation) i sjajnosti boje (value). Prednost HSV formata se očituje u većoj intuitivnosti spektra. Dok kod RGB spektra svaka boja je određena sa tri parametra red, green i blue, kod HSV spektra samo prvi parametar hue određuje boju i ta vrijednost predstavlja kut zakreta na cilindru te stoga

poprima vrijednost od 0 do 355 što označava puni krug. U openCV biblioteci hue (nijansa) vrijednost poprima vrijednosti od 0 do 179, znači upola manje. Idući parametar HSV spektra je saturation (zasićenje) koji se označava kao dominantnost same boje. Na samom obodu cilindra gdje je saturation maksimalna su najdominantnije boje. Treća vrijednost HSV spektra je value (svjetlina). Ona predstavlja os koja prolazi kroz središte kružnice i na svom maksimumu predstavlja crnu boju.

Tablica 1. Odnos percepcijskih i kolorimetrijskih veličina boje

Percepcijska veličina	Kolorimetrijska veličina
Nijansa	Dominantna valna duljina
Zasićenje	Čistoća pobude
Svjetloća	Intenzitet
Sjajnost	Intenzitet

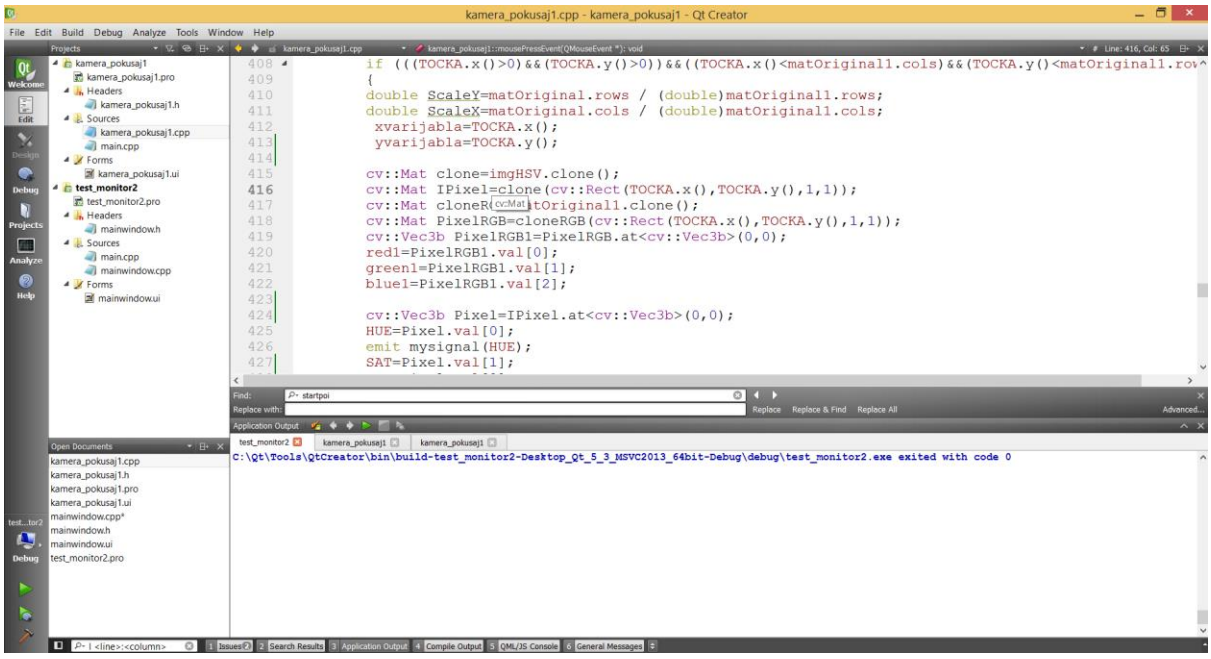
HSV prikaz boja omogućava jednostavno razdvajanje same dominantne valne duljine koju predstavlja nijansa (hue) od intenziteta boje i čistoće pobude koje predstavljaju zasićenje (saturation) i svjetloća (value) te je sama vrijednost nijanse jedan od ključnih parametara u programu.

4. PROGRAMSKO OKRUŽENJE

Glavni i pomoćni program pisana su u Qt programskom okruženju u C++ programskom jeziku te korištenjem njihovih dostupnih biblioteka i vanjske biblioteke za računalni vid OpenCV.

4.1 Qt

Qt je programsko okruženje za razvoj aplikacija koje se mogu pokretati na raznim platformama sa malo ili bez promjena u samom kodu programa. Qt se prvenstveno koristi za razvoj aplikacija sa grafičkim sučeljem. Qt u osnovi koristi standardni C++ programski jezik sa dodatkom signala i slotova koji olakšavaju upravljanje događajima što značajno olakšava razvoj aplikacije. Qt omogućava i pisanje koda u pythonu i javascriptu te omogućava kompiliranje aplikacija za većinu mobilnih i stolnih platformi.



Slika 5. Qt programsko okruženje

4.2 OpenCV

OpenCV je biblioteka otvorenog koda za računalni vid. Ima sučelje za C++, C, Python i Javu i podržava Windowse, Mac OS, iOS i Android. Sama biblioteka sadrži preko 2500 algoritama koji se mogu koristiti u različitim primjenama. Korišteni su samo mali dijelovi biblioteke potrebni za ovaj zadatak u Qt softverskom paketu i C++ programskom jeziku. Korištene funkcije se nalaze u tri biblioteke podataka:

- opencv_core249.dll
- opencv_highgui249.dll
- opencv_imgproc249.dll

5. OPIS GLAVNOG PROGRAMA

5.1 Uvod

Svrha glavnog programa je traženje eMIR robota na njihovom radnom prostoru. Program je izveden na način da uključivanjem programa pretraži sve kamere instalirane na sustav kao i sve COM portove. Kamera i COM port se odabiru iz padajućih izbornika te se svakoj kameri postavlja rezolucija 640x480 kao početna vrijednost koja se može mijenjati upisivanjem željene rezolucije.

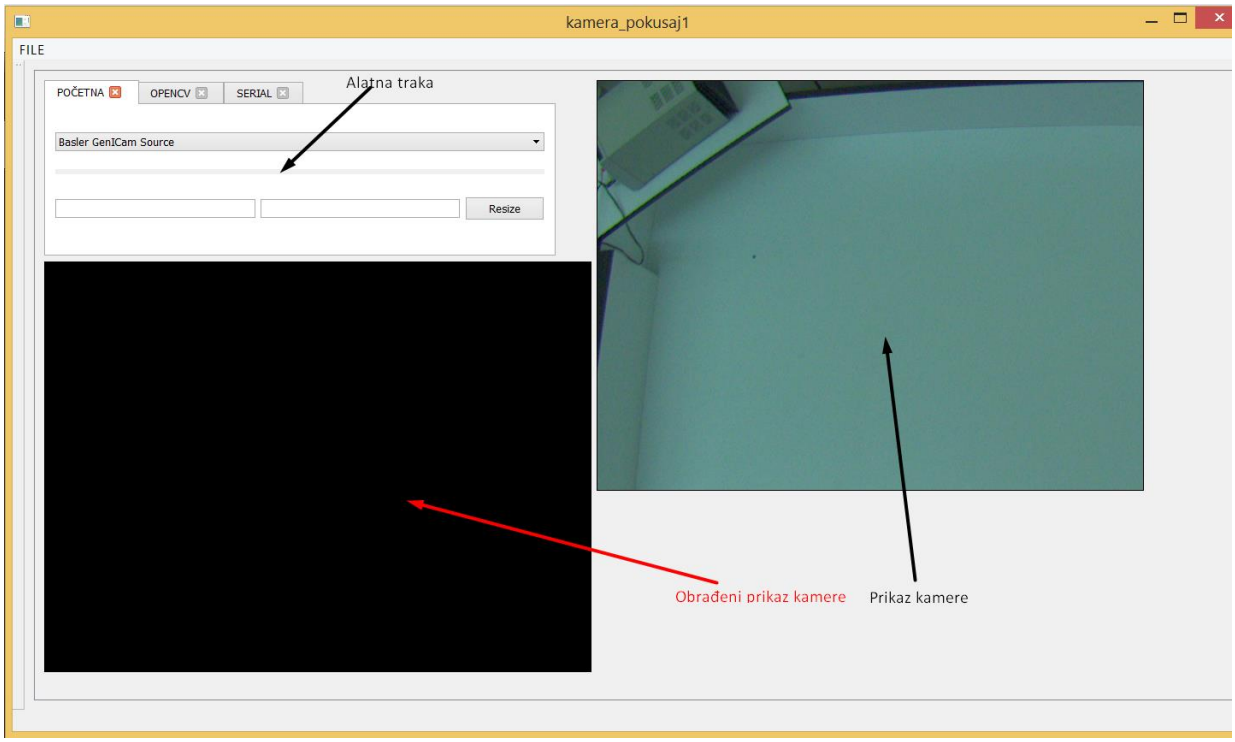
Sam koncept programa je:

1. Akvizicija slike s odabrane kamere
2. Priprema slike za pretraživanje
3. Traženje zadane boje
4. Traženje konture oko zadane boje
5. Filtriranje kontura
6. Izračun položaja konture
7. Priprema podataka za slanje
8. Primanje i slanje podataka

5.2 Sučelje

Sučelje [Slika 6] je izvedeno u programskom paketu Qt te se sastoji od prozora koji je podijeljen na 3 dijela:

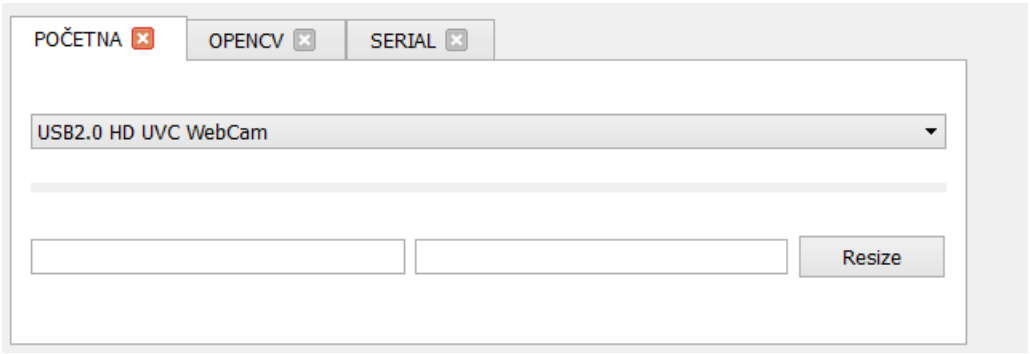
- Alatna traka sa 3 kartice
- Prikaz kamere
- Obrađen prikaz kamere



Slika 6. Sučelje

5.2.1 Prva kartica

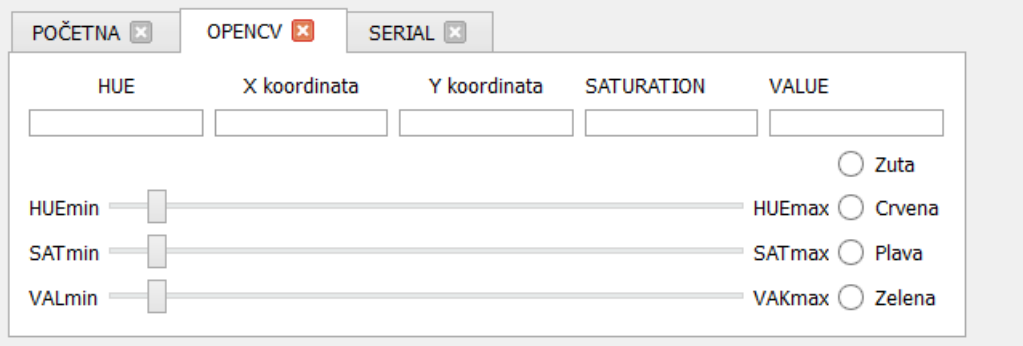
Prva kartica [Slika 7] nosi naziv „POČETNA“ i na njoj se iz padajućeg izbornika odabire kamera te se u dva prazna polja upisuje željena rezolucija. Sve instalirane kamere ubacuju se automatski u izbornik.



Slika 7. Prva kartica

5.2.2 Druga kartica

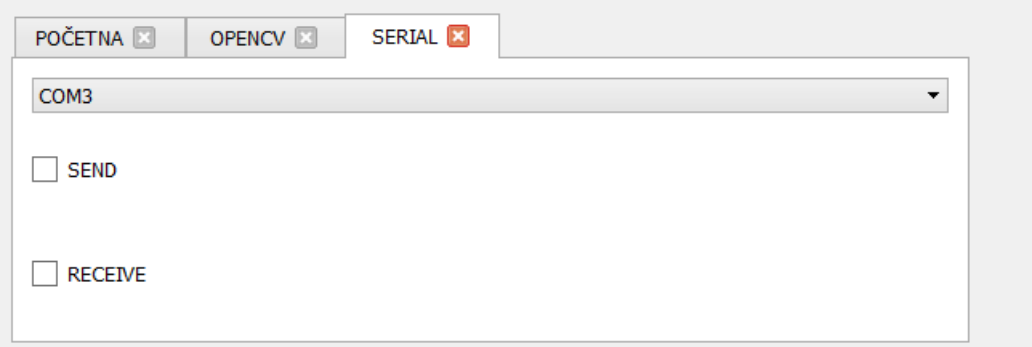
Na drugoj kartici [Slika 8] nalaze se postavke vezane uz filtriranje i snimanje boja. Na vrhu se nalazi 5 praznih polja na kojima se ispisuju vrijednosti nijanse, zasićenje i svjetloće te koordinate težišta. Ispod njih se nalaze tri klizača na kojima se mogu podešavati postavke u slučaju da predefinirane postavke ne zadovoljavaju. Svaki klizač ima raspon od 0 do 80 te maksimalni raspon filtriranja prema tome može biti 160 za svaku od tri vrijednosti. Predefinirana postavke klizača su na 5, što znači da je raspon 10, tj. ±5 od referentne vrijednosti.



Slika 8. Druga kartica

5.2.3 Treća kartica

Na trećoj kartici [Slika 9] nalaze se postavke vezane uz serijski COM port. Kod uključivanja programa COM portovi dodaju se automatski u padajući izbornik te je potrebno odabrati željeni port. Ispod se nalaze dvije tipke od kojih tipka „SEND“ započinje sa slanjem na odabrani port, a tipka „RECEIVE“ sa primanjem sa odabranog porta.



Slika 9. Treća kartica

5.3 Akvizicija slika

Akvizicija slike sa kamere se obavlja deset puta u sekundi i izvedena je na način da timer odbrojava 100 milisekundi te se na kraju svakog odbrojavanja spoji sa funkcijom koja dohvaća sliku s kamere u zadanoj rezoluciji i obrađuje je prema zadanim parametrima. Timer se kreira kao instanca klase QTimer u Qt-u koja emitira signal (timeout) kada odbroji interval:

```
Timer=new QTimer(this);
connect(Timer, SIGNAL(timeout()),this,SLOT(processFrameAndUpdateGUI()));
Timer->start(100);
```

gdje je:

- Timer=new QTimer(this); - stvaranje instance klase QTimer
- connect(Timer, SIGNAL(timeout()),this,SLOT(processFrameAndUpdateGUI())); - funkcija koja spaja Timer sa slotom (funkcijom) koji dohvaća sliku s kamere
- Timer->start(100); - pokretanje odbrojavanja timera s vremenom odbrojavanja od 100 ms.

Slika se dohvaća sa funkcijama:

```
cap.set(CV_CAP_PROP_FRAME_WIDTH, Xres);
cap.set(CV_CAP_PROP_FRAME_HEIGHT, Yres);
cap.read(matOriginal);
```

gdje je:

- cap.*set*(CV_CAP_PROP_FRAME_WIDTH, Xres) - podešavanje parametra širine slike
- cap.*set*(CV_CAP_PROP_FRAME_HEIGHT, Yres) – podešavanje parametra visine slike
- cap.*read*(matOriginal) – akvizicija slike s kamere i spremanje u spremnik matOriginal

5.4 Priprema slike

Nakon što je slika spremljena u memoriju pristupa se konverziji iz RGB-a u HSV model boja kako bi se moglo obaviti pretraživanje po zadanoj nijansi. OpenCV funkcijom cvtColor se prebaci slika prvo iz opencv modela BGR (blue-green-red) u RGB (red-green-blue), a zatim u HSV model:

```
cv::cvtColor(matOriginal, matOriginal1, CV_BGR2RGB);
cv::cvtColor(matOriginal1, imgHSV, CV_RGB2HSV);
```

gdje je:

- matOriginal – slika spremljena s kamere u BGR formatu
- matOriginal1 – konvertirana slika u RGB formatu
- CV_BGR2RGB – naredba koja označava ulazni i izlazni format slike
- imgHSV – konvertirana slika u HSV formatu
- CV_RGB2HSV – naredba koja označava ulazni i izlazni format slike

5.5 Pretraživanje boje

Pretraživanje boja odvija se na način da program pretraži pripremljenu sliku sa zadanim parametrima boje koji su postavljeni na način da se oko zadane boje uzima određeni raspon vrijednosti kako bi se pokrili svi pikseli na predmetu. U programu se širina tog raspona pretraživanja može podešavati kako bi se osiguralo prepoznavanje predmeta i dobila površina koju sačinjavaju pikseli koji zadovoljavaju uvjete pretraživanja. Pretraživanje se vrši opencv funkcijom inRange. Funkcija inRange je idućeg oblika:

```
cv::inRange(imgHSV, cv::Scalar(low[HUE,SAT,VAL]),
cv::Scalar(high[HUE,SAT,VAL]), imgThresholded);
```

gdje je:

- imgHSV – slika u HSV formatu na kojoj se vrši pretraživanje
- cv::Scalar(low[HUE,SAT,VAL]) - donje vrijednosti zadanih parametara boje
- cv::Scalar(high[HUE,SAT,VAL]) – gornje vrijednosti zadanih parametara boje
- imgThresholded – izlazna slika iz funkcije

Tako obrađena slika sa samo prihvatljivim pikselima se još obrađuje opencv funkcijom threshold koja sliku prebacuje u binarnu gdje svi pikseli koji su zadovoljili pretraživanje boja poprimaju vrijednost 255, tj. postaju bijele boje.

```
cv::threshold(imgThresholded, binary, 140, 255, cv::THRESH_BINARY);
```

gdje je:

- imgThresholded – slika na kojoj se vrši obrada
- binary – izlazna binarna slika
- parametar vrijednosti 140 – označava prag iznad kojeg svi pikseli poprimaju maksimalnu vrijednost
- parametar 255 – označava maksimalnu vrijednost parametra koja za 255 znači da svi pikseli vrijednosti intenziteta veće od 140 poprimaju vrijednost 255 što je bijela boja
- cv::THRESH_BINARY – opcija za konverziju u binarnu sliku

5.6 Traženje konture

Nakon pretraživanja boja i nakon što je dobivena binarna slika idući korak u raspoznavanju objekta je pretraživanje kontura oko svih piksela na slici. Opencv funkcija predviđena za to je findcontours. Ta funkcija uzima binarnu sliku i traži konture prema algoritmu[7] i sprema ih kao vektore točaka u niz.

```
cv::findContours(binary, contours, hierarchy, CV_RETR_EXTERNAL,
 CV_CHAIN_APPROX_SIMPLE);
```

gdje je:

- binary – binarna slika na kojoj se traže konture

- contours – predstavlja pronađene konture spremljene u niz
- hierarchy – dodatne informacije o svakoj konturi
- CV_RETR_EXTERNAL – opcija kojom se definira način spremanja kontura. U ovom slučaju spremaju se samo ekstremno vanjske konture
- CV_CHAIN_APPROX_SIMPLE – opcija kojom se definira aproksimacijska metoda kojom se povezuju točke u konturu. Uključena je opcija kojom se sažimaju svi suvišni segmenti i spremaju samo krajnje točko, npr. Pravokutna kontura je kodirana sa samo četiri točke

5.7 Filtriranje kontura

Sve konture spremljene u niz dodatno se filtriraju i obrađuju u petlji koja ima cilj izbaciti sve suvišne konture i one premale koje ne bi mogle predstavljati predmet interesa. Petlja se konstruira na način da se iterira kroz sve konture i svaka prolazi kroz filtre. Prva funkcija kojom se obrađuju konture je approxPolyDP koja aproksimira konture pojednostavljenim krivuljama sa zadanom preciznošću.

```
cv::approxPolyDP(cv::Mat(contours[j]), approx, cv::arcLength(
cv::Mat(contours[j]), true) *0.02,  true );
```

gdje je:

- cv::Mat(contours[j]) - Ulazna kontura.
- aaprox - Rezultat aproksimacije. Mora odgovarati tipu ulazne krivulje.
- cv::arcLength(cv::Mat(contours[j]), true) *0.02 - Parametar koji precizira preciznost aproksimacije. U ovom slučaju je postavljen na 2% dopuštenog odstupanja svake aproksimirane točke od originalne točke.
- true - Parametar koji je boolean tipa i koji sa svojom postavkom true definira da aproksimirana krivulja mora biti zatvorena, tj. prva i zadnja točka moraju biti povezane.

Svaka kontura prolazi na test veličine objekta i test konveksnosti. Funkcijom contourArea se izračunava veličina konture, tj. broj piksela koji sadrži. U slučaju da kontura sadrži manje od 50 piksela ili kontura nije konkavna, ta kontura se odbacuje i uzima se nova kontura.

```
if (std::fabs(cv::contourArea(contours[j])) < 50 || !cv::isContourConvex(approx))
    continue;
```

Konture koje zadovolje testove konveksnosti i veličine idu na daljnje testiranje gdje ulaze u novu petlju koja ih uspoređuje na temelju njihove veličine. Svaka kontura uspoređuje se s dosad najvećom. Kontura koja je najveća predstavlja objekt interesa i izračunava se njezin položaj i veličina.

5.8 Izračun položaja objekta

Da bi se dobio položaj objekta kojeg zapravo predstavlja položaj težišta konture koja je odabrana potrebno je izračunati momente nultog i prvog reda konture. Moment nultog reda zapravo predstavlja sumu svih bijelih piksela. Za izračun momenta koristi se opencv funkcija moments kod koje je ulazna varijabla samo zadana kontura.

```
moment=cv::moments(contours[j]);
```

Za izračun težišta objekta, a samim time i njegovog položaja u prostoru koristi se omjer momenata prvog reda i momenta nultog reda.

```
int CentarXkut=moment.m10/moment.m00;
int CentarYkut=moment.m01/moment.m00;
```

gdje varijabla CentarXkut predstavlja udaljenost objekta po x od nulte točke koja je u gornjem lijevom kutu slike, a CentarYkut predstavlja udaljenost objekta po y osi od nulte točke u gornjem lijevom kutu.

Površina objekta je jednostavno vrijednost momenta nultog reda, tj. suma svih bijelih piksela.

```
double Area=moment.m00
```

5.9 Priprema i format podataka za slanje

Svaki podatak koji se izračuna moraju biti prebačeni u osam bitni format i zapisani pomoću dva broja u heksadekadskom brojevnom sustavu. Sam format u kojem mora biti niz podataka je oblika

*hhxxyyffnnrrCS/

Gdje su :

- hh = predstavlja vrijednost boje
- xx = x koordinata težišta
- yy = y koordinata težišta
- ff = kut koji trenutno predstavlja vrijednost 0
- nn = veličina objekta
- rr = rezerva (za sada 0)
- CS = kontrolni broj

U slučaju da je bilo koja od vrijednosti predstavljena jednoznamenkastim brojem na prvo mjesto se u tom slučaju stavlja 0 pošto svaki član niza mora obavezno sadržavati dva znaka. Prva vrijednost koja se stavlja u niz je vrijednost boje koja je određena zadanim brojem.

Tablica 2. Vrijednosti parametra „hh“ za pojedine boje

boja	broj
Crvena	1
Žuta	3
Zelena	4
Plava	6
Ne prati boju	0

Broj boje koja se šalje pretvara se u heksadekadski broj i u tip podataka char.

```
QString s=QString::number(HUE/2,16).toUpper();
s= s.rightJustified( 2, '0' );
```

Gdje se sa QString::number(HUE/2,16).toUpper() dekadski broj pretvara u heksadekadski.

Drugi član niza je pozicija po x osi. Potrebno je prvo prebaciti vrijednost po x osi u udaljenost od središta.

```
int Xcentar=-Xres/2+CentarXkut;
```

Pošto je rezolucija kamere 1624x1232, to znači da od središta na svaku stranu maksimalna pozicija iznosi 812, a zbog ograničenja na osam bitne podatke najveća vrijednost koja se šalje mora biti 255, te je dogovoreno da se pozicija od središta dijeli sa brojem 8. Pretvorba podataka se radi na istovjetan način kao i sa bojom.

```
QString s1=QString::number(xx/8,16).toUpper();
s1=s1.rightJustified(2,'0');
```

Na istovjetan način se vrši konverzija y koordinate.

```
QString s2=QString::number(yy/8,16).toUpper();
s2=s2.rightJustified(2,'0');
```

Iduća varijabla za slanje predstavlja kut koji je u ovom trenutku nemoguće izračunati zbog oblika eMIR-a koji nema jednoznačno definirane značajke oblika preko kojih bi bilo moguće izračunati kut. Zato se trenutno uzima vrijednost 0.

Peti član niza podataka je vrijednost površine koja je zapravo suma svih piksela koji se nalaze u konturi objekta. Zbog svoje potencijalne veličine površina se dijeli sa 81 te zatim testira za slučaj da li je veća ili manja od 127. U slučaju da je veća dijeli se dodatno sa 9 i dodaje joj se 113 te zatim pretvara u heksadekadski broj.

```
double Area=moment.m00/81;
if(Area<127)
{
        nn=Area;
}
else
{
        nn=(Area/9)+113;
}
QString s3=QString::number(nn,16).toUpper();
s3=s3.rightJustified(2,'0');
```

Šesti član niza predstavlja rezervu koja trenutno nema namjenu pa se uzima vrijednost 0.

Posljednji član niza je kontrolni broj koji je razlika vrijednosti višekratnika broja 256 i sume vrijednosti svih članova niza. Kod kontrolnog broja za konverziju se uzimaju samo dvije posljednje znamenke neovisno o veličini istog.

```
QString s4=QString::number(sum,16).toUpper();

if(sum>256)
        s4=s4.right(2);
s4=s4.rightJustified(2,'0');
```

5.10 Primanje i slanje podataka

Primanje podataka započinje odabirom željenog COM porta i pritiskom na tipku „RECEIVE“ kojom započinje primanje podataka. Podaci se obrađuju kako pristižu na port, tj. kada se aktivira signal da se u ulaznom spremniku nalaze podatci. Serijska veza se otvara sa standardnim postavkama [Tablica 3] .

Tablica 3. Postavke serijske veze

BaudRate	57600
DataBits	Data8
Parity	NoParity
StopBits	OneStop
FlowControl	NoFlowControl

Svi pristigli podaci se spremaju u niz te zatim analiziraju sa dogovorenima naredbama [Tablica 4]. Dogovorene naredbe su:

Tablica 4. Naredbe koje program prima preko COM porta

Naredba	Boja koju prati
#C01/	Crvena
#C03/	Žuta
#C04/	Zelena
#C06/	Plava
#C00/	Ne prati boju

Slanje podataka započinje odabirom porta i pritiskom na tipku „SEND“ i podatci se šalju deset puta u sekundi.

6. OPIS POMOĆNOG PROGRAMA

6.1 Uvod

Pomoćni program [Slika 10] je također pisan u Qt programskom okruženju u C++ programskom jeziku. Napravljen je jednostavno samo sa namjerom prikaza rezultata kojeg primi preko 103 COM porta. Program ima samo osnovni prozor [Slika 10] na koji je učitana slika radnog prostora robota slikana s kamere iznad robota. Program nacrta kružić na mjestu gdje se nalazi robot na temelju podataka koje dobije preko COM porta.

Slika 10. Pomoćni program

6.2 Primanje podataka i konverzija podataka

Pomoćni program ima samo mogućnost primanja podataka preko COM porta 103 sa standardnim postavkama kao u 5.10 na [*Tablica 3. Postavke serijske veze*]. Njegov glavni zadatak je u ulaznom spremniku izdvojiti niz podataka te ga rastaviti i uzeti koordinate te ih pretvoriti u cjelobrojni tip podataka.

6.3 Označavanje položaja

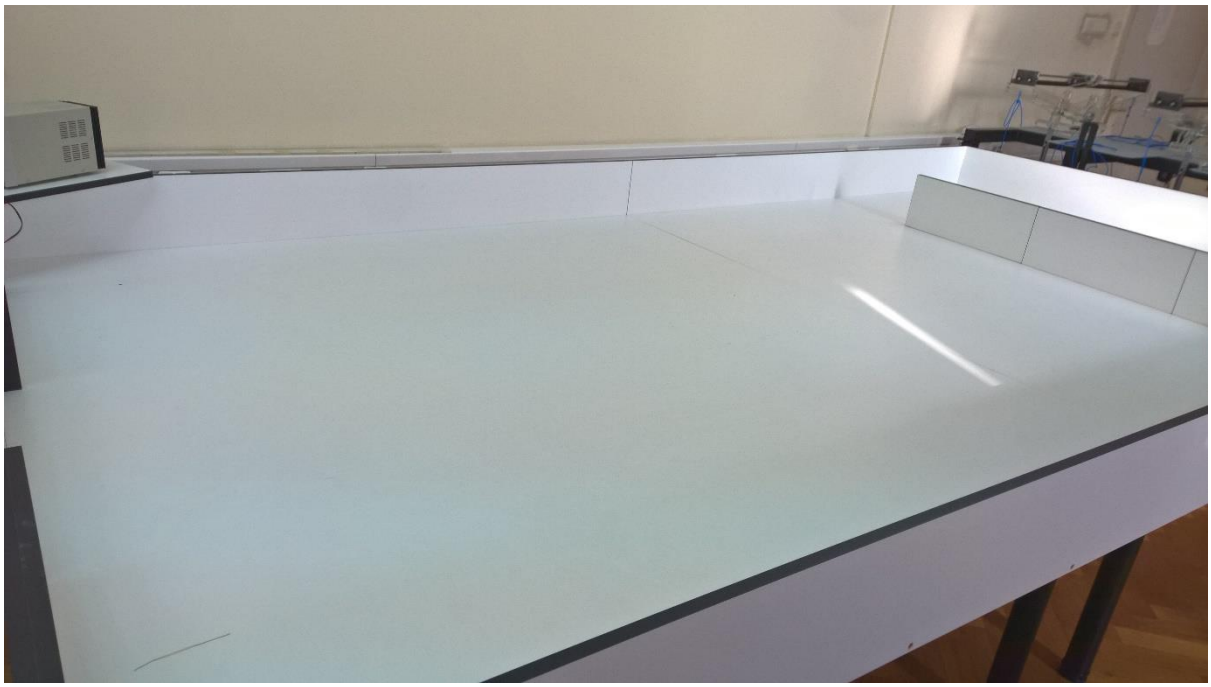
Za pozadinsku sliku na pomoćnom programu uzeta je slika radnog prostora s kamere.

U slučaju uspješnog primanja podataka i konverzije u cjelobrojni tip podataka dobiveni položaj se prikaže u obliku kruga u boji predmeta na zadanim koordinatama na slici u pomoćnom programu.

7. TESTIRANJE PROGRAMA

7.1 Uvod

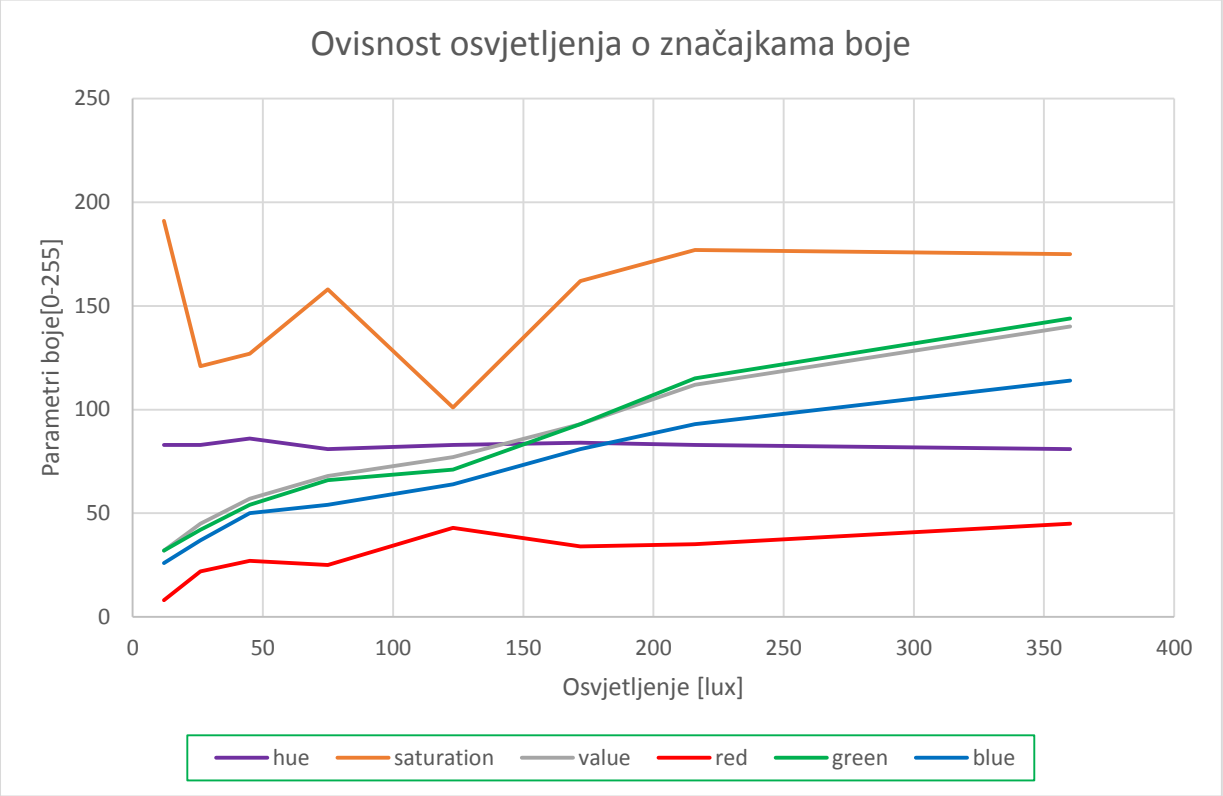
Testiranje je provedeno na dostupnom eMIR-u te na referentnim valjcima crvene, plave, žute i zelene boje. Testiranje se provodilo na radnom prostoru robota [Slika 11] u laboratoriju za automatiku i robotiku 3 na Fakultetu strojarstva i brodogradnje. Mjerenja osvjetljenja provedena su mobilnom aplikacijom te se brojčane vrijednosti ne mogu uzeti kao sasvim pouzdane. Osvjetljenje se sastoji od dnevnog svjetla i halogenih svjetiljki.



Slika 11. Radni prostor robota

7.2 Utjecaj osvjetljenja na značajke boje

Provedeno je ispitivanje utjecaja svjetlosti na značajke boje. Najniža vrijednost osvjetljenja na testu je 12 lux-a dok je najveća 360 lux-a. Na grafu [*Slika 12*2] vidi se utjecaj osvjetljenja na pojedini model boja (HSV,RGB). Usprkos jačem rasipanju zasićenja boje (saturation) HSV model je bolji za korištenje u vizijskim sustavima zbog veće stabilnosti najbitnijeg parametra nijanse (hue).



Slika 12. Utjecaj osvjetljenja na značajke boje

7.3 Test detekcije u uvjetima različitog osvjetljenja

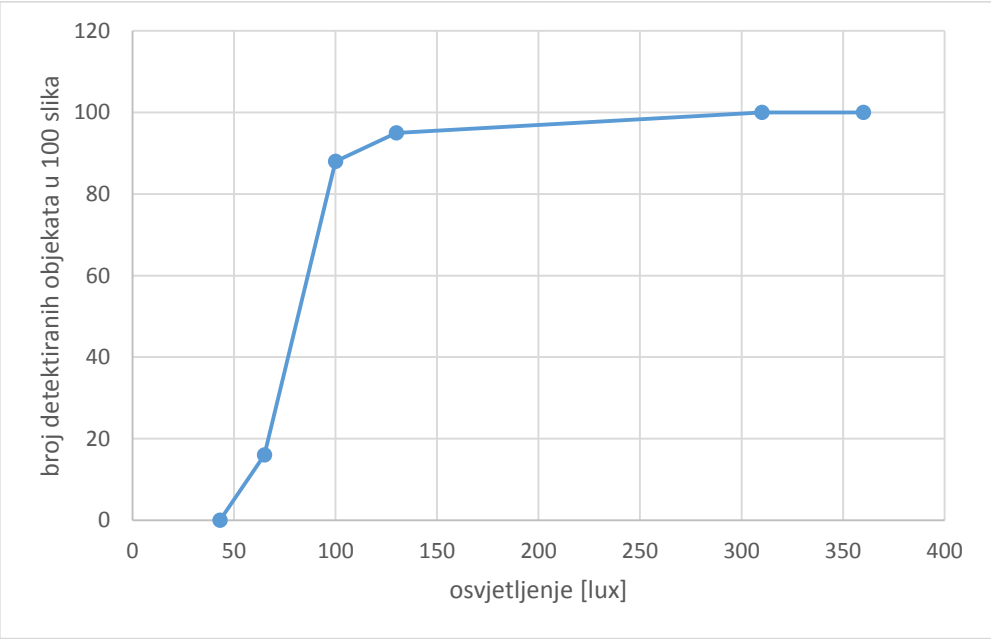
Provedena su dva testa detekcije u uvjetima različitog osvjetljenja. Cilj je bio detektirati plavi predmet s vrijednostima:

- HUE=100
- SAT=55
- VAL=155

Svaki krug testiranja sastoji se od detekcije istog objekta u 100 slika te se promatra koliko je puta predmet detektiran.

7.3.1 Prvi slučaj

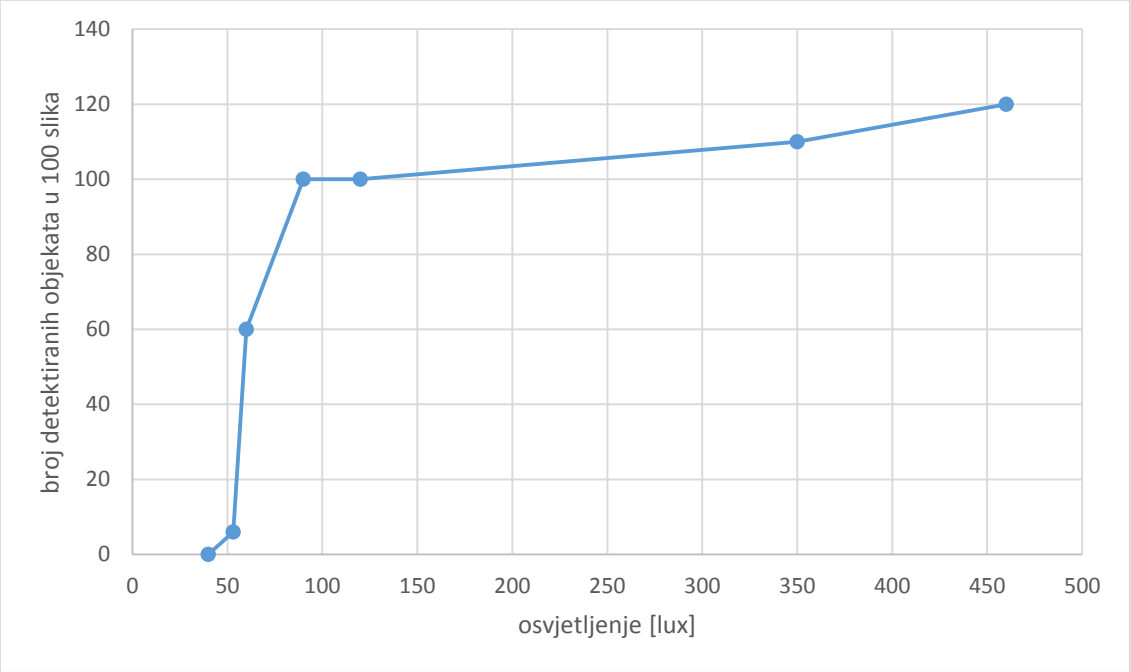
Program je baždaren prema uvjetima maksimalnog osvjetljenja u vrijednosti od 360 lux-a za prvi slučaj te je na toj vrijednosti proveden i prvi krug testiranja. U programu je raspon HSV vrijednosti podešen na ±40 od referentne vrijednosti. Na slici [Slika 13] vidimo kako program detektira objekt u 100% slučajeva na maksimalnim vrijednostima osvjetljenja te kako uspješnost detekcije pada polagano do osvjetljenja 100 lux-a te pripadajuće detekcije od 88 objekata. Minimum detekcija ostvaren je na vrijednosti od 65 lux-a za ovaj slučaj kad je detektirano 16 objekata. Na nižim vrijednostima nije detektiran niti jedan objekt.



Slika 13. Test detekcije(parametri ±40)

7.3.2 Drugi slučaj

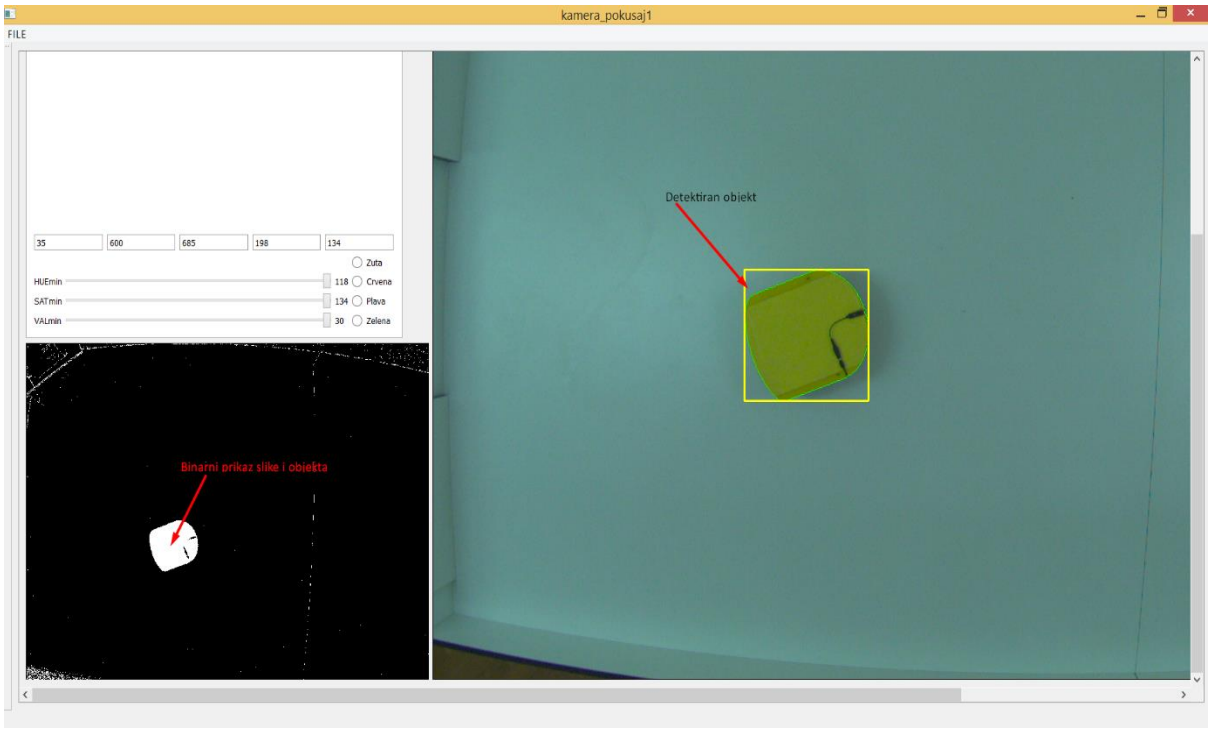
U ovom slučaju program je također baždaren na maksimalnom osvjetljenju koje je dosegnulo 460 lux-a. Test je obavljen na istom plavom objektu jednakih vrijednosti kao i u prvom slučaju. Raspon HSV vrijednosti podešen je na ±80 od referentnih vrijednosti te predstavlja maksimum. Na slici [Slika 14] se vidi posljedica tako širokog raspona vrijednosti. Na nekim razinama osvjetljenja program je detektirao više objekata nego što je imao slika. Zbog širokog raspona polja vrijednosti došlo je do pojave šuma te je program isti predmet detektirao više puta u istoj slici. Pojavom šuma predmet je izgubio jasne konture te je program pronašao više varijacija iste konture koje su zadovoljile sve uvjete testiranja te ih je pogrešno detektirao. Iako nije poželjno da se predmet detektira u više bliskih kontura to zbog malenih varijacija između svake od njih ne predstavlja problem.



Slika 14. Test detekcije(parametri ±80)

7.4 Testiranje

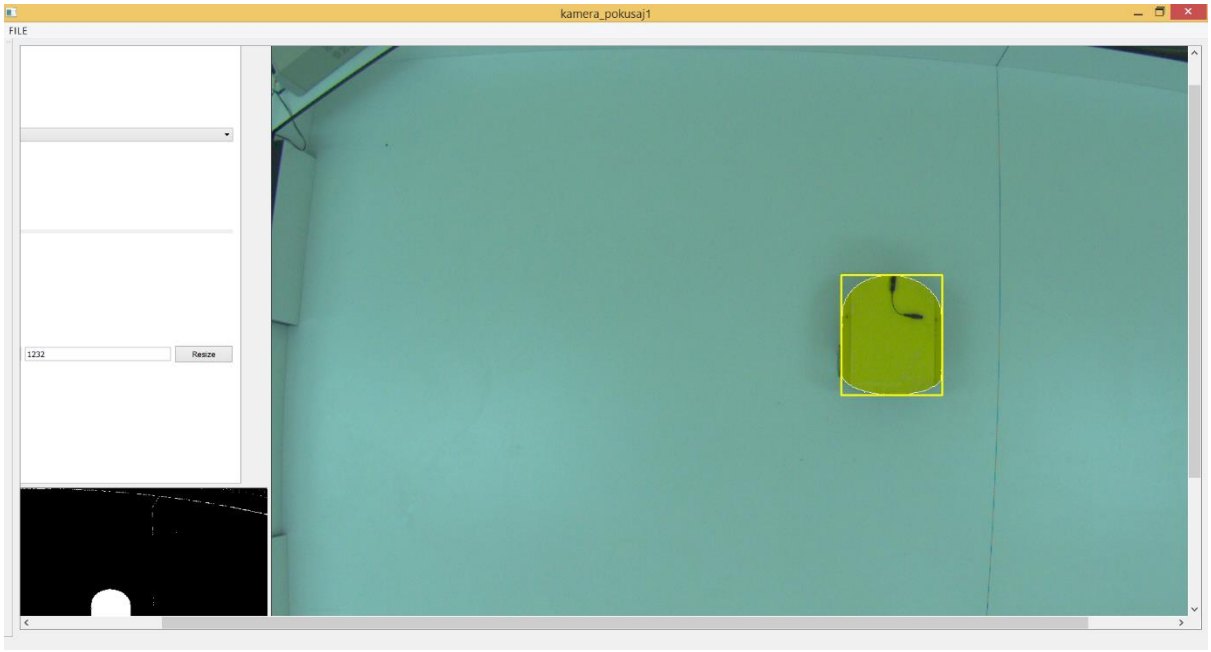
Testiranje se odvijalo otvaranjem virtualnog komunikacijskog kanala programom Virtual Serial Ports Emulator. Kreirani su virtualni COM portovi 102 i 103. Na virtualni port 102 se dogovorno spaja glavna aplikacija koja traži robote i šalje informacije o položaju, a na virtualni port 103 pomoćna aplikacija koja prikaže položaj robota. Učenje boje se odvija pritiskom lijeve tipke miša na željenu boju na slici te zatim pritiskom na gumb kraj imena boje na alatnoj traci. Potom se odabire COM port preko kojeg će se odvijati komunikacija te pritiskom na tipke „SEND“ i „RECEIVE“ program je spreman za obostranu komunikaciju. Nakon primanja zapovijedi o praćenju boje program žutim opisnim pravokutnikom oko objekta na prikazu kamere pokazuje da je objekt te boje pronađen [Slika 15] te šalje podatke na COM port.



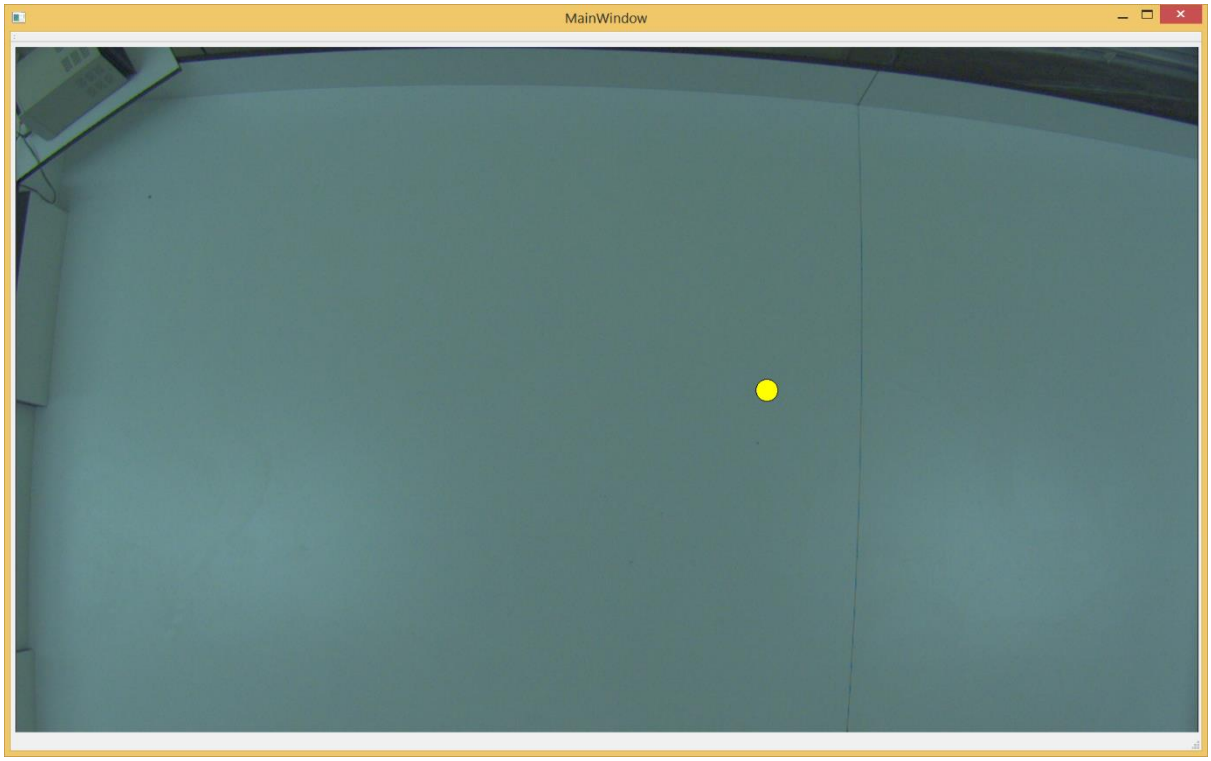
Slika 15. Detekcija objekta

Na slici [Slika 16] vidi se objekt detektiran u glavnom programu dok je na slici [Slika 17] objekt prikazan u pomoćnom programu krugom prema informacijama koje je primio preko COM porta. Iz slike [Slika 16] i slike [Slika 17] vidi se da lokacija eMIR-a na pomoćnom programu odgovara lokaciji eMIR-a na glavnom programu što pokazuje uspješnost detekcije

samog objekta, komunikacije preko virtualnog kanala te uspješnost primanja tih informacija s pomoćnim programom.



Slika 16. Test programa



Slika 17. Primljeni položaj i prikazan u pomoćnom programu

8. Zaključak

Testiranje je pokazalo da program nakon učenja boja radi u konstantnim uvjetima te prepoznaje eMIR robota u svakom slučaju. Program može razlikovati eMIR robota od ostatka okoliša na temelju razlike između boje eMIR-a i okoliša. Kretanje robota se može pratiti u realnom vremenu zahvaljujući izračunu i slanju pozicije svakih 100 ms. Program se može prilagoditi različitim svjetlosnim uvjetima kroz učenje boja, ali ti uvjeti nakon učenja moraju ostati konstantni. U slučaju promjene svjetlosnih uvjeta detekcija postaje otežana. Za najbolje rezultate detekcije potrebno je parametre prilagoditi trenutnim svjetlosnim uvjetima. Također, uzimanje osnovnih informacija kao što su položaj i veličina objekta radi na svakom detektiranom objektu. Komunikacija između programa i prikaz rezultata odvija se preko COM portova te je uspješno realizirana komunikacijom između glavnog i pomoćnog programa. Pomoćni program dobiva ispravne podatke preko komunikacijskog kanala te ih prikazuje. Program omogućava posredništvo između više programa sa svojim mogućnostima komunikacije te će omogućiti razvoj drugih programa koji će dohvaćati informacije kroz njega.

9. LITERATURA

[1] Crneković M., Zorc D. i Kunica Z., Research of mobile robot behaviour with eMIR, IN-TECH 2012, Rijeka

[2] http://lab405.fesb.hr/igraf/Frames/fP5_1.htm, 2015-11-15.

[3] https://en.wikipedia.org/wiki/RGB_color_model, 2015-11-15.

[4] Šribar J., Motik B., Demistificirani C++, Element, Zagreb, 2010.

[5] http://docs.opencv.org/2.4/doc/tutorials/tutorials.html, 2015-11-15.

[6] http://doc.qt.io/qt-5/qtexamplesandtutorials.html, 2015-11-15.

[7] Suzuki, S., Abe, K.: Topological Structural Analysis of Digitized Binary Images by Border Following, 1985.

[8] Ryannel J. i Thelin J., QT5 Cadaques, [e-book], 2015, https://qmlbook.github.io/assets/qt5_cadaques.pdf, 2015-10-4.

[9] https://www.youtube.com/watch?v=0ONxIy8itRA, 2015-10-15.

[10] http://opencv-srf.blogspot.hr/2010/09/object-detection-using-color-seperation.html, 2015-9-28

[11] http://doc.qt.io/qt-4.8/signalsandslots.html, 2015-10-15.

[12] http://doc.qt.io/qt-5/qtserialport-terminal-example.html, 2015-10-15.

[13] http://docs.opencv.org/2.4/modules/core/doc/operations_on_arrays.html#inrange, 2015-10-15.

[14] http://docs.opencv.org/master/d4/d73/tutorial_py_contours_begin.html#gsc.tab=0, 2015-10-15

[15] http://stackoverflow.com/questions/22470902/understanding-moments-function-in-opencv, 2015-10-15

[16] http://stackoverflow.com/questions/22132510/opencv-approxpolydp-for-edge-maps-not-contours, 2015-10-15

PRILOZI

I. CD sa programom
II. Računalni kod glavnog programa

```
//////////////////////////////PRO FILE////////////////////////////////////

QT       += core gui
QT       +=multimedia multimediawidgets
QT       +=serialport
greaterThan(QT_MAJOR_VERSION, 4): QT += widgets

TARGET = kamera_pokusaj1
TEMPLATE = app


SOURCES += main.cpp\
        kamera_pokusaj1.cpp

HEADERS  += kamera_pokusaj1.h

FORMS    += kamera_pokusaj1.ui

INCLUDEPATH += C:\\Users\\spanja\\Documents\\opencv\\opencv\\build\\include

LIBS += -LC:\\Users\\spanja\\Documents\\opencv\\opencv\\build\\x64\\vc12\\lib \
    -lopencv_calib3d249 \
    -lopencv_contrib249 \
    -lopencv_core249 \
    -lopencv_features2d249 \
    -lopencv_flann249 \
    -lopencv_gpu249 \
    -lopencv_highgui249 \
    -lopencv_imgproc249 \
    -lopencv_legacy249 \
    -lopencv_ml249 \
    -lopencv_nonfree249 \
    -lopencv_objdetect249 \
    -lopencv_ocl249 \
    -lopencv_photo249 \
    -lopencv_stitching249 \
    -lopencv_superres249 \
    -lopencv_ts249 \
    -lopencv_video249 \
    -lopencv_videostab249 \
//////////////////////////////////////////////////////////////////////////
```

```
//////////////////////header/////////////////////////////////////////

#ifndef KAMERA_POKUSAJ1_H
#define KAMERA_POKUSAJ1_H

#include <QtMultimedia/QCameraInfo>
#include <QtMultimedia/QCamera>
#include <QtMultimedia/QMediaRecorder>
#include <QMainWindow>
#include<QtCore>
#include <opencv2/opencv.hpp>
#include <opencv2/core/core.hpp>
#include <opencv2/highgui/highgui.hpp>
#include <opencv2/imgproc/imgproc.hpp>
#include <QGraphicsScene>
#include <QString>
#include <QGraphicsView>
#include <QGraphicsTextItem>
#include <QtSerialPort/QtSerialPort>
#include <QMouseEvent>
#include <QGraphicsSceneMouseEvent>
#include <QPointer>
#include <QLabel>



namespace Ui {
class kamera_pokusaj1;
}

class kamera_pokusaj1 : public QMainWindow
{
    Q_OBJECT

public:
    explicit kamera_pokusaj1(QWidget *parent = 0);
    ~kamera_pokusaj1();

//////////////////////////////////////////////////
    Ui::kamera_pokusaj1 *ui;
    cv::Mat imgHSV;
    cv::Mat imgThresholded;
    cv::Moments moment;
    QString FPS;
    QPoint TOCKA;
    int IndexKonture;
    cv::Point2f TOCKA1;
    int xvarijabla;
    int yvarijabla;
    int HUE, SAT, VAL;
```

```
    int red1,green1,blue1;
    int ValueSlider;
    QSerialPort *serial;
    cv::VideoCapture cap;
    cv::Mat matOriginal, matOriginal1;
    QPixmap image, image1,pixmap;
    QImage  Qoriginal;
    QCamera camera;
    int BKamera;
    QPointer<QGraphicsTextItem> error;
    int pos_x;
    int pos_y;
    QPointer<QTimer> Timer;
    QPointer<QTimer> Timer1;
    QPointer<QTimer> Timer2;
    int i=1;
    int X,Y;
    int Xres,Yres;
    QString t;
    QPoint startPoint;
    QPoint mouseCursorPos;
    bool drawBox;
    QRect *box;


//////////////////////////////////////////////////

signals:

    void mysignal(int BojaClick);

//////////////////////////////////////////////////

private:

    QByteArray serialData;
    QByteArray sendSerialBuffer;
    QString serialBuffer;
    QString serialfiltered;
    int HUEgreen,HUEred,HUEblue,HUEyellow;
    int SATgreen,SATred,SATblue,SATyellow;
    int VALgreen,VALred,VALblue,VALyellow;

//////////////////////////////////////////////////

protected:

     void mousePressEvent(QMouseEvent *ev);

//////////////////////////////////////////////////
```

```
private slots:
    void serialread();
    void on_comboBox1_activated(const QString &arg1);
    void serialwrite();
    void on_pushButton_2_clicked();
    void on_serialOpcija_clicked(bool checked);
    void on_GreenColor_clicked(bool checked);
    void on_RedColor_clicked(bool checked);
    void on_BlueColor_clicked(bool checked);
    void on_YellowColor_clicked(bool checked);
    void on_comboBoxSerial_activated(int index);
//////////////////////////////////////////////////
public slots:
    void processFrameAndUpdateGUI();
    void setLabel(cv::Mat& im, const std::string label, std::vector<cv::Point>& contour);
    double angle(cv::Point pt1, cv::Point pt2, cv::Point pt0);
    void NoCamera();


};
#endif // KAMERA_POKUSAJ1_H

//////////////////////////////////////////////////////////////////

//////////////////////GLAVNI PROGRAM////////////////////////////////////////////////////

#include "kamera_pokusaj1.h"
#include "ui_kamera_pokusaj1.h"
```

```
Q_DECLARE_METATYPE(QCameraInfo)
kamera_pokusaj1::kamera_pokusaj1(QWidget *parent) :
   QMainWindow(parent),
   ui(new Ui::kamera_pokusaj1)
{
   ui->setupUi(this);

   X=640;
   Y=480;



serial =new QSerialPort(this);
serialBuffer= "";
 Timer2=new QTimer(this);
 QList<QSerialPortInfo> serialPortInfoList = QSerialPortInfo::availablePorts();
 foreach (const QSerialPortInfo &serialPortInfo, serialPortInfoList)
 {
    ui->comboBoxSerial->addItem(serialPortInfo.portName());
 }

//////////////popunjavanje kamera u padajući izbornik///////////////////////////////////

ui->kamera->installEventFilter(this);
   QActionGroup *videoDevicesGroup = new QActionGroup(this);

   foreach (const QCameraInfo &cameraInfo, QCameraInfo::availableCameras()) {
      QAction *videoDeviceAction = new QAction(cameraInfo.description(),
videoDevicesGroup);
      videoDeviceAction->setCheckable(true);
      videoDeviceAction->setData(QVariant::fromValue(cameraInfo));
      if (cameraInfo == QCameraInfo::defaultCamera())
         videoDeviceAction->setChecked(true);
       ui->comboBox1->addItem(cameraInfo.description());
}
////////////////////////////////////////////////////////////////////////////////

//////////////////////padajući izbornik, aktivacija, kamera//////////////////////////
kamera_pokusaj1::on_comboBox1_activated(t);

   Timer=new QTimer(this);
   connect(Timer, SIGNAL(timeout()),this,SLOT(processFrameAndUpdateGUI()));
   Timer->start(100);
////////////////////////////////////////////////////////////////////////////////

}


//////////////////////destructor/////////////////////////////////////
```

```
kamera_pokusaj1::~kamera_pokusaj1()
{
   delete ui;
   serial->close();
}


//////////////////////////////////////////////////////


/////////////////funkcija slanja//////////////////////
void kamera_pokusaj1::serialwrite()
{
   QString s=QString::number(HUE/2,16).toUpper(); //konverzija u hex
   s= s.rightJustified( 2, '0' );
   int CentarXkut=moment.m10/moment.m00;
   int CentarYkut=moment.m01/moment.m00;
   int Xcentar=-Xres/2+CentarXkut;
   int Ycentar=Yres/2-CentarYkut;
   int xx=Xcentar;
   int yy=Ycentar;
   int nn;

   ui->Xkoordinata->setText(QString::number(xx));
   ui->Ykoordinata->setText(QString::number(xx));


   QString ff="00";
   double Area=moment.m00/81;
   if(Area<127)
   {
      nn=Area;
   }
   else
   {
      nn=(Area/9)+113;
   }
   QString rr="00";
   QString s1=QString::number(xx/8,16).toUpper();  //konverzija xx polozaja
   int hexxx,hexyy;
   if(xx<0)
   {
      s1=s1.right(2);
      hexxx=256+(xx/8);
   }
   s1=s1.rightJustified(2,'0');
   QString s2=QString::number(yy/8,16).toUpper();  //konverzija yy polozaja
   if(yy<0)
   {
      s2=s2.right(2);
```

```
    hexyy=256+(yy/8);
  }
  s2=s2.rightJustified(2,'0');
  QString s3=QString::number(nn,16).toUpper();  //konverzija povrsine
  s3=s3.rightJustified(2,'0');


///////////////izračun CS//////////////////////////
  int CS;
  int sum=-1;
  int mnozitelj=1;
  if ((xx<0)&&(yy<0))
    {
      CS=hexxx+hexyy+HUE/2+nn;
      while(sum<0)
        {

        sum=(mnozitelj*256)-CS;
        mnozitelj++;

        }
    }
  else if((xx<0)&&(yy>0))
    {
      CS=hexxx+yy/8+HUE/2+nn;
      while(sum<0)
        {

        sum=(mnozitelj*256)-CS;
        mnozitelj++;

        }
    }
  else if((xx>0)&&(yy<0))
    {
      CS=xx/8+hexyy+HUE/2+nn;
      while(sum<0)
        {

        sum=(mnozitelj*256)-CS;
        mnozitelj++;

        }
    }
  else
    {
      CS=xx/8+yy/8+HUE/2+nn;
      while(sum<0)
        {
```

```
            sum=(mnozitelj*256)-CS;
            mnozitelj++;

            }
        }

////////////////////////////////////////////////
    QString s4=QString::number(sum,16).toUpper();  //konverzija cs u hex

    if(sum>256)
       s4=s4.right(2);
    s4=s4.rightJustified(2,'0');
///////////konstrukcija niza za slanje//////////////////////////////
    sendSerialBuffer.append("*");
    sendSerialBuffer.append(s);
    sendSerialBuffer.append(s1);
    sendSerialBuffer.append(s2);
    sendSerialBuffer.append(ff);
    sendSerialBuffer.append(s3);
    sendSerialBuffer.append(rr);
    sendSerialBuffer.append(s4);
    sendSerialBuffer.append("/");


    serial->write(sendSerialBuffer);
    sendSerialBuffer.clear();

}

////////////////////////////////////////////////////////

//////////////////////primanje preko seriala////////////////
void kamera_pokusaj1::serialread()
{

    serialData=serial->readAll();
    QString serialData1=serialData;
    serialBuffer += serialData1;
    //ui->lineEdit->setText(QString(serialData1));

QStringList bufferSplit= serialBuffer.split("#");
    serialfiltered=bufferSplit[1];


if(serialfiltered=="C04/")  //zelena boja
    {

        HUE=HUEgreen;
        SAT=SATgreen;
        VAL=VALgreen;
```

```
        serialBuffer="";
}
else if(serialfiltered=="C01/")  //crvena boja
    {

    HUE=HUEred;
    SAT=SATred;
    VAL=VALred;
    serialBuffer="";
    }

else if(serialfiltered=="C03/")  //zuta boja
    {
    HUE=HUEyellow;
    SAT=SATyellow;
    VAL=VALyellow;
    serialBuffer="";
    }

else if(serialfiltered=="C06/")  // plava boja
    {

    HUE=HUEblue;
    SAT=SATblue;
    VAL=VALblue;
    serialBuffer="";
  }

else if(serialfiltered=="C00/")  //niti jedna boja
    {
    HUE=0;
    SAT=0;
    VAL=0;
    }

else
    {
        serialBuffer="";
    }

}

//////////////////////////////////////////////////////



//////////////////////glavna funkcija////////////////////////////

void kamera_pokusaj1::processFrameAndUpdateGUI()
{
```

```
   cap.set(CV_CAP_PROP_FRAME_WIDTH, Xres);
   cap.set(CV_CAP_PROP_FRAME_HEIGHT, Yres);
   cap.read(matOriginal);
   if(matOriginal.empty()==true){kamera_pokusaj1::NoCamera();}
   else
      {
//////////////////prebacivanje u HSV//////////////////////////

   cv::cvtColor(matOriginal, matOriginal1, CV_BGR2RGB);
   cv::Size size(X,Y);
   cv::cvtColor(matOriginal1, imgHSV, CV_RGB2HSV);

 ////////////////////////////////////////////////////////////

 ////////////////////pretraživanje boje//////////////////////

   cv::inRange(imgHSV,cv::Scalar(HUE-ui->HUEminSlider->value(),SAT-ui->SATslider-
>value(),VAL-ui->VALslider->value()),
         cv::Scalar(HUE+ui->HUEminSlider->value(),SAT+ui->SATslider->value(),VAL+ui-
>VALslider->value()),imgThresholded);
   cv::Mat binary;
   cv::threshold(imgThresholded, binary, 140, 255, cv::THRESH_BINARY);
            std::vector<std::vector<cv::Point>> contours;
            std::vector<cv::Vec4i>hierarchy;
            cv::findContours(binary, contours, hierarchy, CV_RETR_EXTERNAL,
CV_CHAIN_APPROX_SIMPLE);
            cv::Rect bounding_rect;

////////////////////////////////////////////////////////////

//////////////////////pretrazivanje kontura//////////////////

std::vector<cv::Point> approx;

for( int j=0; j<contours.size();j++)
{
   cv::approxPolyDP(cv::Mat(contours[j]), approx,
                  cv::arcLength(cv::Mat(contours[j]), true) * 0.02,
                  true
               );

                // preskace male konture
    if (std::fabs(cv::contourArea(contours[j])) < 50 || !cv::isContourConvex(approx))
         continue;

////////////////////////////////////////////////////////////////////////

//////////////////////trazenje dovoljno velike konture//////////////////////////
    if(ui->SerialPracenje->isChecked())
    {
```

```
        cv::Rect bounding_rect;        //definiranje opisnog pravokutnika
        int largest_area = 0;   int largest_contour_index = 0;
        for (int i = 0; i< contours.size(); i++)
           {
           double a = contourArea(contours[i], false);
           if (a>largest_area)
                {
                    largest_area = a;
                    largest_contour_index = i;
                    bounding_rect = boundingRect(contours[i]);
                }
            }
////////////////////////////////////////////////////////////////////////////

    cv::Scalar color(255, 255, 255);   //crtanje konture pomoću drawContours funkcije na slici filtriranoj prema
    drawContours(imgThresholded, contours, largest_contour_index, color, CV_FILLED, 8, hierarchy);
    moment=cv::moments(contours[j]);
    cv::drawContours(matOriginal1, contours, j, color, 1, 8, hierarchy); //crtanje konture na slici
    cv::rectangle(matOriginal1, bounding_rect,  cv::Scalar(255,255,0),2, 8,0);

    }

    if (!ui->SerialPracenje->isChecked())
           {
           double DobraKontura=cv::pointPolygonTest(contours[j],cv::Point2f(xvarijabla,yvarijabla), false);
           cv::Scalar color(0, 255, 0);
           if(DobraKontura>0)
                    {
                     bounding_rect=cv::boundingRect(contours[j]);
                     moment=cv::moments(contours[j]);
                     if (approx.size() >= 4 && approx.size() <= 6)
                         {
                             // broj stranica
                             int vtc = approx.size();
                             std::vector<double> cos;
                             for (int m = 2; m < vtc+1; m++)
                             cos.push_back(angle(approx[m%vtc], approx[m-2], approx[m-1]));

                             // sortiranje kutova
                             std::sort(cos.begin(), cos.end());

                             // najnizi i najveci kut
                             double mincos = cos.front();
                             double maxcos = cos.back();

                             // oblik konture
```

```
                        if (vtc == 4 && mincos >= -0.1 && maxcos <= 0.3)
                            setLabel(matOriginal1, "RECT", contours[j]);
                    }

                    cv::drawContours(matOriginal1, contours, j, color, 1, 8, hierarchy);
                    cv::rectangle(matOriginal1, bounding_rect,  cv::Scalar(255,255,0),2, 8,0);
                    ui->label_2->setText(QString::number(bounding_rect.width));
                }
                }
}
  QImage Qoriginal((uchar*)matOriginal1.data, matOriginal1.cols, matOriginal1.rows,
matOriginal1.step, QImage::Format_RGB888);
image = QPixmap::fromImage(Qoriginal);
 ui->kamera->setScaledContents(false);
ui->kamera->setPixmap(image);
cv::resize(imgThresholded,imgThresholded,size,0,0,0);
QImage QInrange((uchar*)imgThresholded.data, imgThresholded.cols,
imgThresholded.rows, imgThresholded.step, QImage::Format_Indexed8);
image1 = QPixmap::fromImage(QInrange);
ui->obrada->setPixmap(image1);

   }

}

//////////////////pisanje po slici////////////////////////////////////////////////
void kamera_pokusaj1::setLabel(cv::Mat& im, const std::string label,
std::vector<cv::Point>& contour)
{
   int fontface = cv::FONT_HERSHEY_SIMPLEX;
   double scale = 0.4;
   int thickness = 1;
   int baseline = 0;

   cv::Size text = cv::getTextSize(label, fontface, scale, thickness, &baseline);
   cv::Rect r = cv::boundingRect(contour);

   cv::Point pt(r.x + ((r.width - text.width) / 2), r.y + ((r.height + text.height) / 2));
   cv::rectangle(im, pt + cv::Point(0, baseline), pt + cv::Point(text.width, -text.height),
CV_RGB(255,255,255), CV_FILLED);
   cv::putText(im, label, pt, fontface, scale, CV_RGB(0,0,0), thickness, 8);
}

////////////////////////////////////////////////////////////////////////////

//////////////////////////izracun kuta između stranica/////////////////////
double kamera_pokusaj1::angle(cv::Point pt1, cv::Point pt2, cv::Point pt0)
{
   double dx1 = pt1.x - pt0.x;
   double dy1 = pt1.y - pt0.y;
```

```
   double dx2 = pt2.x - pt0.x;
   double dy2 = pt2.y - pt0.y;
   return (dx1*dx2 + dy1*dy2)/sqrt((dx1*dx1 + dy1*dy1)*(dx2*dx2 + dy2*dy2) + 1e-10);
}
////////////////////////////////////////////////////////////////////////////

//////////////////////left mouse click za uzimanje hue ////////////////////////
void kamera_pokusaj1::mousePressEvent(QMouseEvent *ev)
{

   if(ev->button()==Qt::LeftButton)
   {

      startPoint=ev->pos();
      box=new QRect(startPoint.x(),startPoint.y(),0,0);
      drawBox=true;
      ui->kamera->setScaledContents(false);

       TOCKA= ui->kamera->mapFrom(this,startPoint);


      if
(((TOCKA.x()>0)&&(TOCKA.y()>0))&&((TOCKA.x()<matOriginal1.cols)&&(TOCKA.y(
)<matOriginal1.rows)))
      {
      double ScaleY=matOriginal.rows / (double)matOriginal1.rows;
      double ScaleX=matOriginal.cols / (double)matOriginal1.cols;
       xvarijabla=TOCKA.x();
       yvarijabla=TOCKA.y();

      cv::Mat clone=imgHSV.clone();
      cv::Mat IPixel=clone(cv::Rect(TOCKA.x(),TOCKA.y(),1,1));
      cv::Mat cloneRGB=matOriginal1.clone();
      cv::Mat PixelRGB=cloneRGB(cv::Rect(TOCKA.x(),TOCKA.y(),1,1));
      cv::Vec3b PixelRGB1=PixelRGB.at<cv::Vec3b>(0,0);
      red1=PixelRGB1.val[0];
      green1=PixelRGB1.val[1];
      blue1=PixelRGB1.val[2];

      cv::Vec3b Pixel=IPixel.at<cv::Vec3b>(0,0);
      HUE=Pixel.val[0];
      emit mysignal(HUE);
      SAT=Pixel.val[1];
      VAL=Pixel.val[2];
      ui->label_2->setText(QString::number(SAT));
      ui->HueObjava->setText(QString::number(HUE));
      ui->label_10->setText(QString::number(VAL));

}
```

```
    }
}
///////////////slucaj da nema kamere/////////////////
void kamera_pokusaj1::NoCamera()
{


ui->obrada->setText("NO CAMERA");
    ui->kamera->setText("NO CAMERA");
}
//////////////////////////////////////////////////////

//////////////////kamera izbornik aktivacija////////////////
void kamera_pokusaj1::on_comboBox1_activated(const QString &arg1)
{
    cap.open(ui->comboBox1->currentIndex());
    if (cap.isOpened()==false)
    {

        kamera_pokusaj1::NoCamera();

    }
}
//////////////////////////////////////////////////////

/////////////////promjena rezolucije//////////////////
void kamera_pokusaj1::on_pushButton_2_clicked()
{

    ui->lineEdit1->setInputMask("9999");
    ui->lineEdit2->setInputMask("9999");

    Xres=ui->lineEdit1->text().toInt();
     Yres=ui->lineEdit2->text().toInt();
     if (Xres==0){Xres=640;}
     if (Yres==0){Yres=480;}

}
//////////////////////////////////////////////////////

/////////////slanje preko serial porta////////////////////////
void kamera_pokusaj1::on_serialOpcija_clicked(bool checked)
{
    if(ui->serialOpcija->isChecked())
    {
    Timer1->start(100);
    connect (Timer1,SIGNAL(timeout()),this,SLOT(serialwrite()) );
    }
    else
    {
```

```
        Timer1->stop();;
    }
}
//////////////////////////////////////////////////////

//////////////////zelena boja naredba//////////////////////
void kamera_pokusaj1::on_GreenColor_clicked(bool checked)
{

    HUEgreen=HUE;
    SATgreen=SAT;
    VALgreen=VAL;


}
//////////////////////////////////////////////////////////

//////////////////crvena boja naredba//////////////////////
void kamera_pokusaj1::on_RedColor_clicked(bool checked)
{
    HUEred=HUE;
    SATred=SAT;
    VALred=VAL;
}
//////////////////////////////////////////////////////

//////////////////plava boja naredba//////////////////
void kamera_pokusaj1::on_BlueColor_clicked(bool checked)
{
    HUEblue=HUE;
    SATblue=SAT;
    VALblue=VAL;
}
//////////////////////////////////////////////////////

//////////////zuta boja naredba//////////////////////////
void kamera_pokusaj1::on_YellowColor_clicked(bool checked)
{
    HUEyellow=HUE;
    SATyellow=SAT;
    VALyellow=VAL;
}
//////////////////////////////////////////////////////

/////////////////odabir serijskog porta i otvaranje/////////
void kamera_pokusaj1::on_comboBoxSerial_activated(int index)
{
    serial->setPortName(ui->comboBoxSerial->currentText());
    serial->open(QIODevice::ReadWrite);
```

```
    serial->setBaudRate(QSerialPort::Baud57600);
    serial->setDataBits(QSerialPort::Data8);
    serial->setParity(QSerialPort::NoParity);
    serial->setStopBits(QSerialPort::OneStop);
    serial->setFlowControl(QSerialPort::NoFlowControl);

    Timer1=new QTimer(this);

    connect(serial, SIGNAL(readyRead()),this,SLOT(serialread()));
}
```

III. Računalni kod pomoćnog programa

```
//////////////////PRO FILE////////////////////////////////

#-------------------------------------------------
#
# Project created by QtCreator 2015-11-12T12:03:35
#
#-------------------------------------------------

QT       += core gui
QT       +=multimedia multimediawidgets
QT       +=serialport

greaterThan(QT_MAJOR_VERSION, 4): QT += widgets

TARGET = test_monitor2
TEMPLATE = app


SOURCES += main.cpp\
        mainwindow.cpp

HEADERS  += mainwindow.h

FORMS    += mainwindow.ui


//////////////////////////////////////////////////////////////////////////


//////////////////////////HEADER/////////////////////////////////////

#ifndef MAINWINDOW_H
#define MAINWINDOW_H

#include <QMainWindow>
```

```
#include <QtSerialPort/QtSerialPort>
#include<QtCore>
#include <QGraphicsScene>
#include <QString>
#include <QGraphicsView>
#include <QGraphicsTextItem>
#include <QtSerialPort/QtSerialPort>
#include <QMouseEvent>
#include <QGraphicsSceneMouseEvent>
#include <QPointer>
#include <QLabel>

namespace Ui {
class MainWindow;
}

class MainWindow : public QMainWindow
{
   Q_OBJECT

public:
   explicit MainWindow(QWidget *parent = 0);
   QSerialPort *serial;
   QPointer<QTimer> Timer1;
   ~MainWindow();

private:
   Ui::MainWindow *ui;
   QByteArray serialData;
   QString serialBuffer;
   QString serialfiltered;
   QString boja;
   QPixmap pixmap1;
   QPainter painter2;

private slots:
   void serialread();
};

#endif // MAINWINDOW_H


////////////////////////////////////////////////////////////////////////////


/////////////////GLAVNI PROGRAM//////////////////////////////

#include "mainwindow.h"
#include "ui_mainwindow.h"
```

```
MainWindow::MainWindow(QWidget *parent) :
   QMainWindow(parent),
   ui(new Ui::MainWindow)
{
   ui->setupUi(this);

//////////otvaranje porta///////////////

   serial = new QSerialPort(this);
   serial->setPortName("com101");
     serial->open(QIODevice::ReadWrite);
     serial->setBaudRate(QSerialPort::Baud57600);
     serial->setDataBits(QSerialPort::Data8);
     serial->setParity(QSerialPort::NoParity);
     serial->setStopBits(QSerialPort::OneStop);
     serial->setFlowControl(QSerialPort::NoFlowControl);

////////////////////////////////////////////////
QPixmap pixmap("C:/capture.bmp");
ui->label_3->setPixmap(pixmap);
      Timer1=new QTimer(this);
    serialBuffer="";

     connect(serial, SIGNAL(readyRead()),this,SLOT(serialread()));
}
/////////////////citanje porta//////////////////////////

void MainWindow::serialread()
{
   serialData=serial->readAll();
   QString serialData1=serialData;
   serialBuffer += serialData1;

   QStringList bufferSplit= serialBuffer.split("*");
      serialfiltered=bufferSplit[1];
      serialBuffer="";
      if(serialfiltered.size()>17)
         serialfiltered="";

         boja=serialfiltered.left(2);
         bool ok;
      int hexboja=boja.toInt(&ok,16);
      QString xxpos=serialfiltered.mid(2,2);
      int xx=(xxpos.toInt(&ok,16));
      if(xx>130)
         {
            xx=-256+xx;
            xx=xx*8;
         }
      else {
```

```
        xx=xx*8;
       }
       QString yypos=serialfiltered.mid(4,2);
       int yy=(yypos.toInt(&ok,16));
       if(yy>130)
          {
             yy=-256+yy;
             yy=yy*8;
          }
       else {
         yy=yy*8;
       }
       QString velicina=serialfiltered.mid(8,2);
       int pov=velicina.toInt(&ok,16);
       qDebug()<<xx;
       qDebug()<<yy;
        xx=824+xx;
        yy=616-yy;

       serialfiltered="";
       QPoint tocka;
////////////crtanje polozaja/////////////////

   QPixmap pixmap1("C:/capture.bmp");
   QPainter painter2(&pixmap1);
   if(hexboja==6)
{
       painter2.setBrush(QBrush(Qt::blue));
       painter2.drawEllipse(xx,yy,30,30);
}
       else if(hexboja==1)
{
     painter2.setBrush(QBrush(Qt::red));
     painter2.drawEllipse(xx,yy,30,30);
}
   else if(hexboja==4)
{
  painter2.setBrush(QBrush(Qt::green));
  painter2.drawEllipse(xx,yy,30,30);
}
   else if(hexboja==3)
{
  painter2.setBrush(QBrush(Qt::yellow));
  painter2.drawEllipse(xx,yy,30,30);
}
      //crtanje polozaja
      ui->label_3->setPixmap(pixmap1);

}
//////////////////////////////////////
```

```
MainWindow::~MainWindow()
{
   delete ui;
}
```